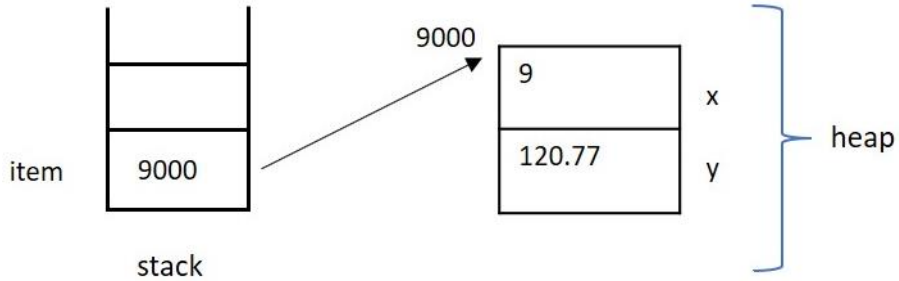
Chapter 1: Understanding Basic C++ Assumptions

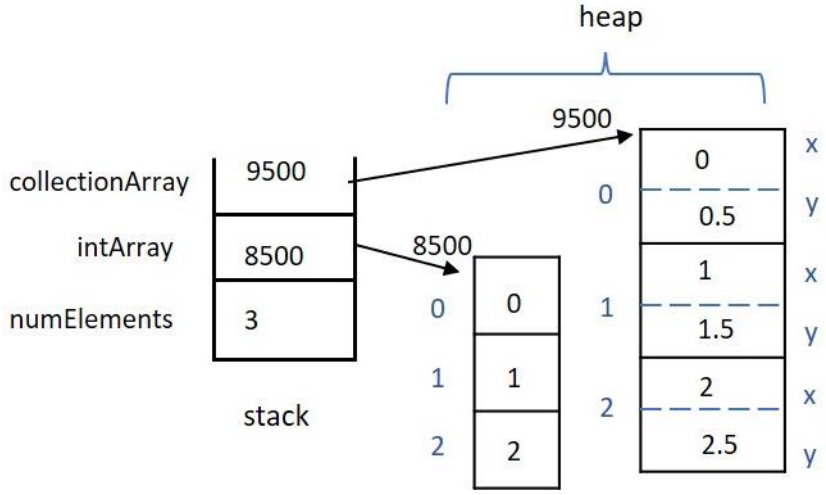
Binary operators			
+	addition	+=	plus equal
-	subtraction	-=	minus equal
*	multiplication, pointer indirection	*=	times equal
/	division	/=	div equal
%	modulus (remainder from an integer divide)		
&	address-of, reference, bitwise AND		
|	bitwise OR		
^	bitwise XOR		
<<	left shift, insertion operator		
>>	right shift, extraction operator		
==	equality (comparison)		
!=	inequality		
<	less than	<=	less than or equals
>	greater than	>=	greater than or equals
&&	logical AND		
||	logical OR		
Unary operators			
+	unary plus		
-	unary minus		
~	one's complement		
!	not		
--	decrement		
++	increment		
Ternary operator			
?:	conditional expression		

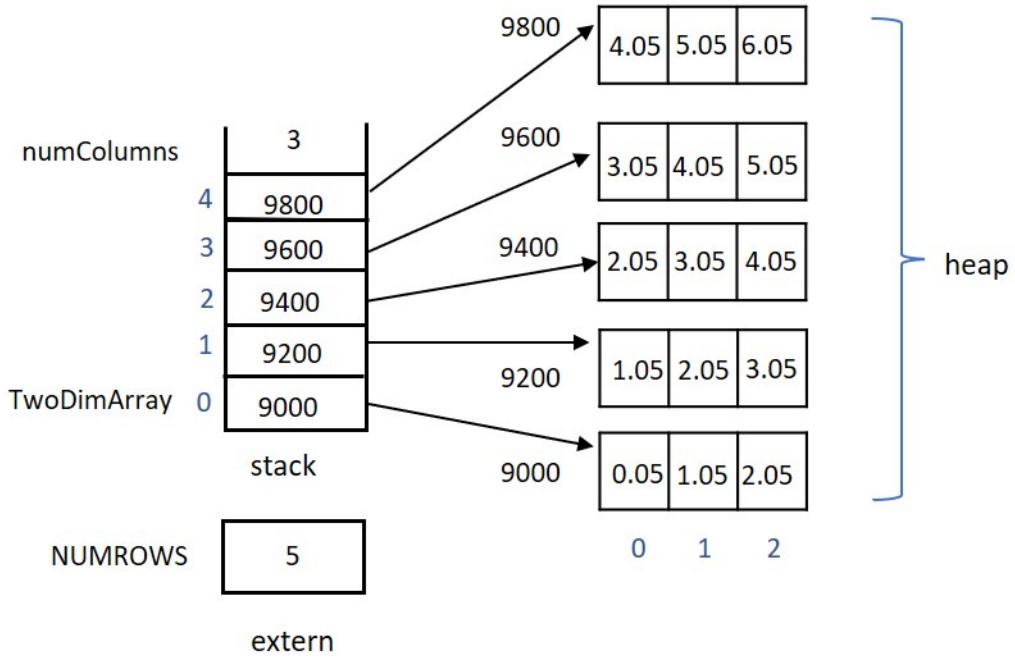
Chapter 2: Adding Language Necessities

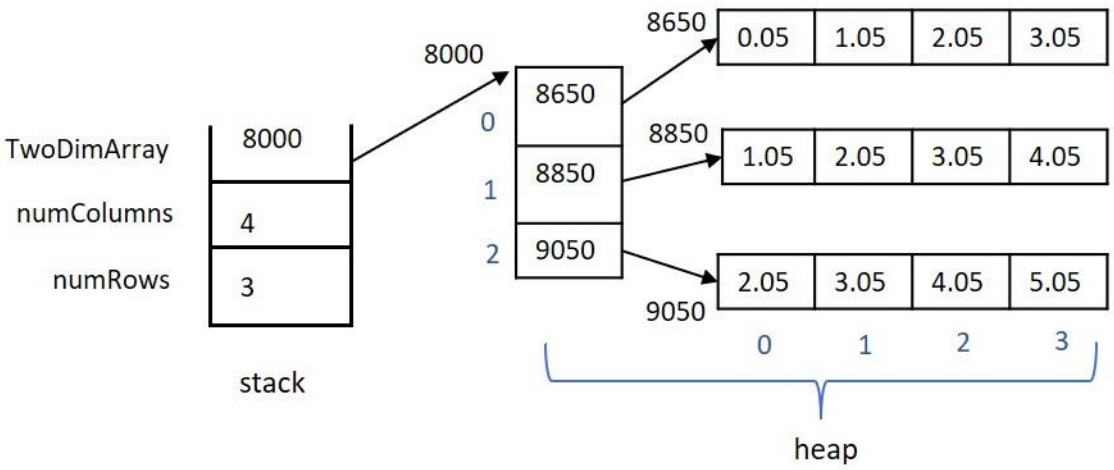
No images…

Chapter 3: Indirect Addressing: Pointers








TwoDimArray
numColumns
numRows
8000
4
3
stack
8000
0
1
2
8650
8850
9050
8650
0.05
1.05
2.05
3.05
8850
1.05
2.05
3.05
4.05
9050
2.05
3.05
4.05
5.05
0
1
2
3
heap

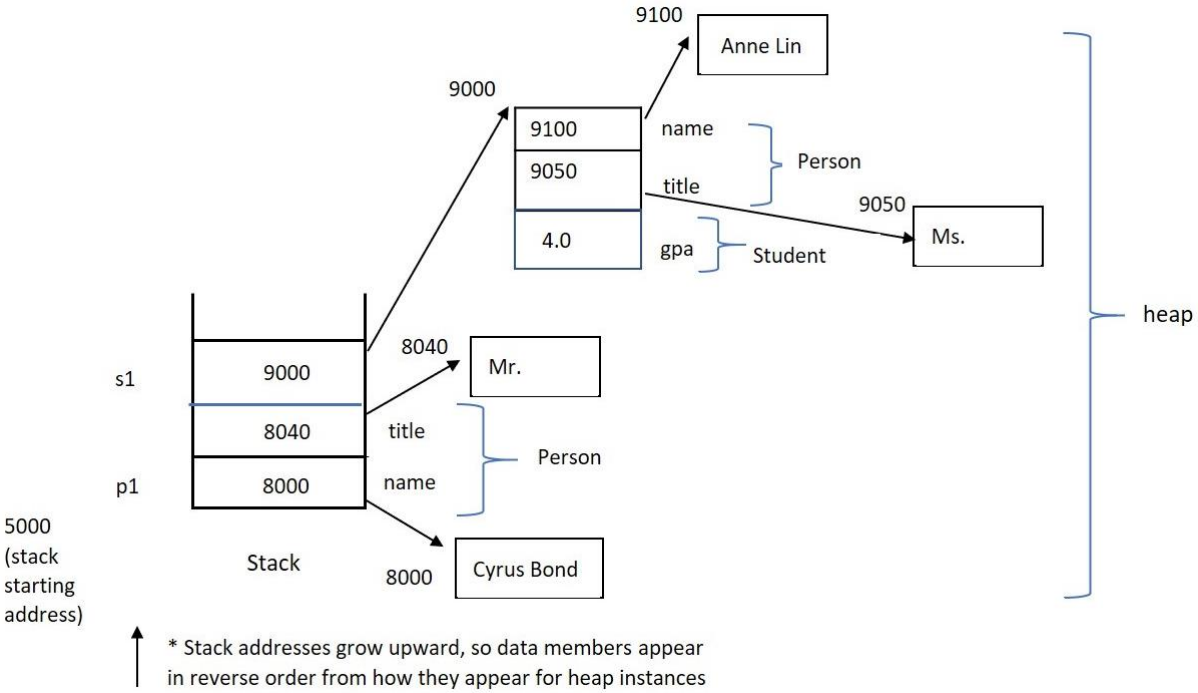
Chapter 4: Indirect Addressing: References

No-images…

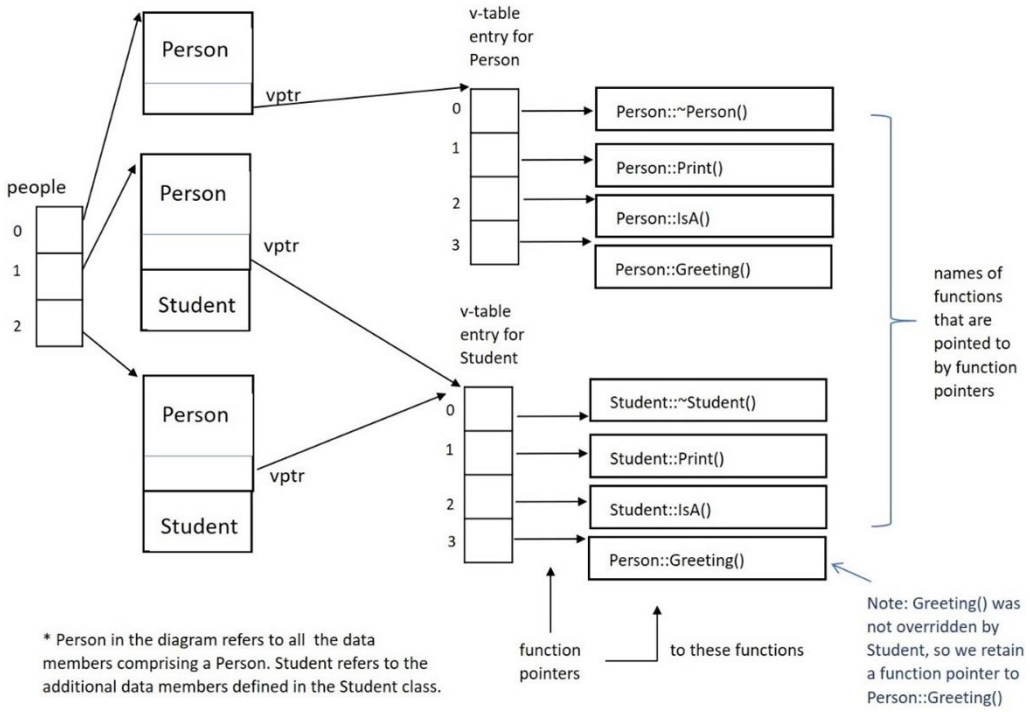
Chapter 5: Exploring Classes in Detail

No-images…

Chapter 6: Implementing Hierarchies with Single Inheritance



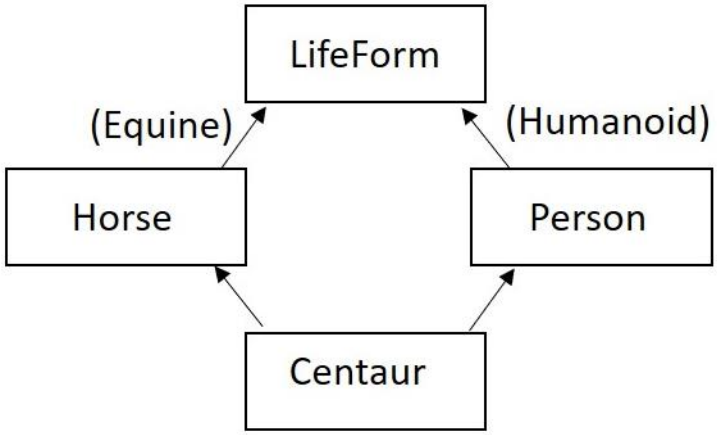
Chapter 7: Utilizing Dynamic Binding through Polymorphism



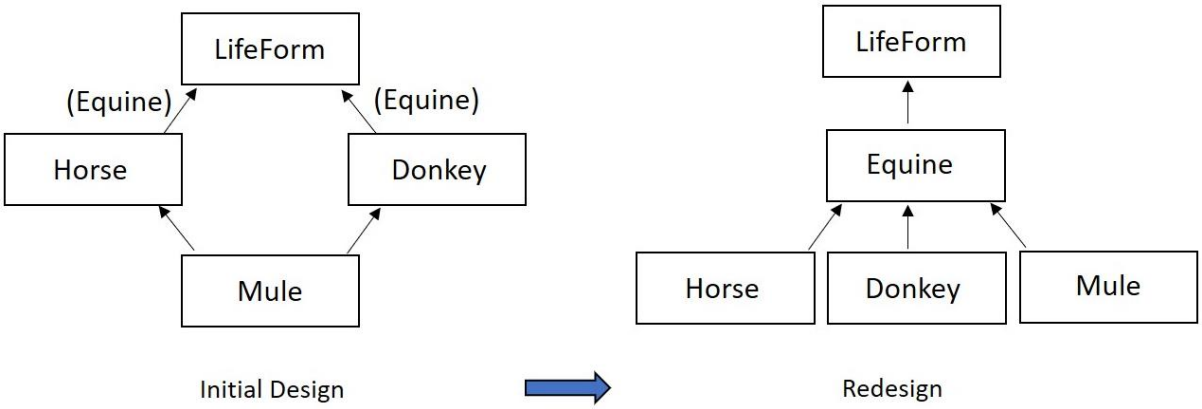
Chapter 8: Mastering Abstract Classes

No-images…

Chapter 9: Exploring Multiple Inheritance



* Discriminators are in ()'s



Chapter 10: Implementing Association, Aggregation, and Composition

No-images…

Chapter 11: Handling Exceptions

No-images…

Chapter 12: Friends and Operator Overloading

No-images…

Chapter 13: Working with Templates

No-images…

Chapter 14: Understanding STL Basics

size_type size() const;	Returns number of elements
void push_back (const value_type &element);	Adds an element to the end
bool empty() const;	Returns TRUE if vector is empty
void clear();	Erases all elements in vector
iterator begin();	Returns an iterator pointing to the first element
iterator end();	Returns an iterator pointing to the last element

iterator begin();	Returns an iterator pointing to the first element
iterator end();	Returns an iterator pointing to the last element
size_type size size() const;	Returns number of elements
void push_back (const value_type &element);	Adds an element to end
void push_front (const value_type &element);	Adds an element to front
void pop_back ();	Removes element from end
void pop_front ();	Removes element at front
bool empty() const;	Returns TRUE if empty
void clear();	Erases all elements

size_type size() const;	Returns number of elements
void push (const value_type&element);	Adds an element to top
void pop();	Removes element from top
bool empty() const;	Returns TRUE if empty
value_type &top();	Returns top element
const value_type &top() const;	Returns top element (read only)

size_type size size() const;	Returns number of elements
bool empty() const;	Returns TRUE if empty
value_type &front();	Returns front element
const value_type &front() const;	Returns front element (read only)
value_type &back();	Returns back element
const value_type &back() const;	Returns back element (read only)
void push (const value_type &element);	Adds element to back of queue
void pop();	Removes element from front of queue

`size_type size() const;`	Returns number of elements
`bool empty() const;`	Returns TRUE if empty
`const value_type &top() const;`	Returns top element (read only)
`void push (const value_type &element);`	Adds an element to back
`void pop();`	Removes element from front
`void emplace(args);`	A new element is constructed using args and then added

`bool empty() const;`	Returns TRUE if empty, FALSE otherwise
`size_type size() const;`	Returns number of elements
`void clear();`	Empties out the map
`size_type erase(const key_type &k);`	Erases element with key is k
`iterator erase(const_iterator pos);`	Erases element at iterator
`void swap(map &);`	Swaps contents of 2 maps
`iterator insert(iterator pos, const value_type &val);`	Insert val at pos
`iterator begin();`	Returns iterator pointing to first element
`iterator end();`	Returns iterator pointing past end
`iterator find(const key_type &k);`	Find element whose key is k

Chapter 15: Testing Classes and Components

No-images…

Chapter 16: Using the Observer Pattern

No-images…

Chapter 17: Applying the Factory Pattern

No-images…

Chapter 18: Applying the Adapter Pattern

No-images…

Chapter 19: Using the Singleton Pattern

No-images…

Chapter 20: Removing Implementation Details Using the pImpl Pattern Final

No-images…

Chapter 21: Making C++ Safer

No-images…